Spotipy – Analyzing the Top Hits

Project Members: Cam Foster, Anis Ali, Michael Alread, Ida Astaneh

Release Date: 2020-10-27

# The Original Data

With so many viable options for the project, our group wanted to focus on a subject that was relevant to our everyday lives that could continue to be analyzed long past the project. Using the tools that we have learned over the last few weeks; our group feels confident that analyzing streaming services such as Spotify can lead to real-life answers to real-life questions.

The Janitors were tasked with finding a large dataset that would provide enough detail to come to a meaningful conclusion using Pandas and Matplotlib principles. The original data comes from Kaggle.com and features with Top 200 songs daily starting on January 1, 2017. The full scope includes actuals from 2017 thru January 2018 across 17 countries worldwide. Given the sheer magnitude of information, our group decided to focus on a specific subset in order to draw those meaningful conclusions without getting “wrapped” up in information.

# Process Overview

To expand our original dataset and get the audio features for each track we needed clean the dataset to filter for only songs in the US region for 2017 extract just the unique track IDs. That allowed us to call the API fewer times and reduce the time it took to get our data.

Once we had the audio features for all songs in our final dataset, we merged the original dataset with the audio features dataset on track ID giving us our final, working dataset.

Next, we began performing data exploration and analysis. The team broke tasks up based on our initial questions and began investigating the dataset for trends and ways to visualize and quantitatively assess the data.

# Cleaning the Data

As mentioned in the ‘Original Data’ subsection, our group took an initial pass thru the downloaded .csv to get a broad idea of any “erroneous” information. Because our data had so many duplicated entries on Artists and Track Names, to ensure data integrity, we referenced the Track IDs to ensure that entries were viable to use. Very similarly to previous homework assignments, we wanted to drop any rows that were missing data or had questionable entries to avoid outliers. Even with the “cleaned” Jupyter notebook, we knew that we had to continue splicing the data in order to help answer any questions that we had hypothesized about. Creating “bins” allowed us to bucket the number of streams in order to see what songs were truly “popular” vs songs that were repetitive on the Top 200 list.



# Spotify API

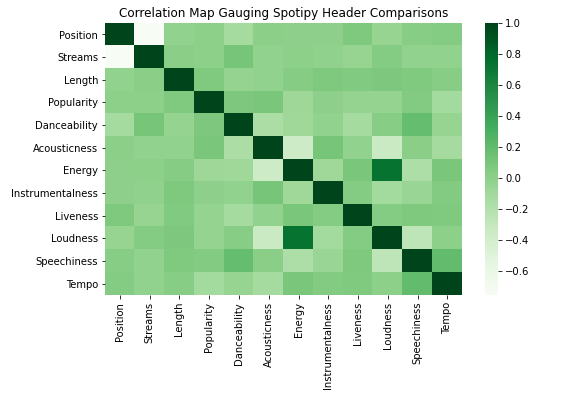
There is a python library called spotipy that is used to interact with the Spotify Web API. You are required to create a developer account and get a Client ID and Client Secret key to authenticate with Spotify. Most of the functions in this library are geared towards interacting with Spotify for web applications and doing things such as selecting the next track, getting related artists, and get current user data. We used two functions, spotipy.track() and spotipy.audio\_features(), to get information about the track requested.

A function was written to call the API and return the track identifiers (i.e. artist, track name, album name, etc.) and audio features (i.e. loudness, acousticness, energy, etc.). We ran into an issue with the API timing out but adding a pause under an except clause gave us a work around for this issue.

All the audio features are defined by Spotify on their developer website.

# Questions and Hypothesis

## Given the data provided, is there a direct correlation between our Spotipy headers?



Not all data headers had a correlation, but the few that did were intuitive. For example, as loudness increases (i.e. the song is overall louder) the energy also increases. This makes sense to anyone who listens to music. If a song is louder there is a sense of more energy. Conversely, if a song is quieter it is more relaxing. An example of two audio features that are inversely correlated would be loudness and acousticness. An acoustic song would primarily be individual or un-amplified instruments, neither of which would create a very loud musical profile.

## Could we bucket the stream counts to show the percentage breakdown? How “popular” is popular?

## Which artist/album appears the most and least amount of times on the list?

After filtering the data down to the US and 2017 only, we recognized that there were still many points of data but at a much more readable level. When looking at the Artist with the most appearances in compared to streams, “Drake” took the top spot. “Phil Collins” checked in with the least amount of streams when compared to the Artist. “Congratulations” by Post Malone fittingly won the award of most streams and highest position. “Give me Love” by Ed Sheeran was streamed the least amount of times in 2017. This Track, however, was released in 2011 which gives the Artist credit for holding down a top 200 spot six years removed.

## What decade, prior to 2010, was responsible for providing the most streams in 2017?

Chart, bar chart

Description automatically generated

To investigate streams and track count by decade, we chose to remove all songs from the 2010s. This is because our main year of interest, 2017, has significantly greater total stream count to the point that previous decades data gets washed out in visualizations and regressions. We binned the unique tracks by decade to create a bar chart for visualization and then performed a linear regression on stream count. The short answer is that the 90s had the most streams prior to the 2010s. However, the linear regression showed that the correlation (R2 value) between Total Stream Count and track Release Year is -0.08, so it is very weakly negatively correlated. Essentially there is no correlation between showing a link between release date and the tendency for a song to show up on a modern Top Track list. We could find no determining factor that would explain why an older song may showed up on our 2017 list.

Chart, scatter chart

Description automatically generated